4. MODEL DEVELOPMENT

This chapter discusses the design of the research model, hereafter referred to as the Mobile Emission Assessment System for Urban and Regional Evaluation (MEASURE). The chapter provides descriptions of the data, processes, algorithms, and files. The chapter will review each of the input files, pre-processing steps, program modules, and program code. By the end of the chapter, the reader should have a clear vision of the model scope and process.

The purpose of the MEASURE is to provide researchers with a tool for measuring the air quality impacts of urban and regional transportation policy and development changes. This model is not intended to be used directly for conformity or inventory reporting, but for use in a research environment by scientists exploring the transportation and air quality relationship.

MEASURE will produce hourly transportation facility-level and gridded automobile exhaust emission estimates of carbon monoxide (CO), hydrocarbons (HC), and oxides of nitrogen (NOx). The model develops these estimates based on a geographic area's vehicle registration data, accurate digital road dataset, TRANPLAN (travel demand forecasting) model output, and zone-based socioeconomic data (i.e., US Census Blocks, Land Use). The outputs of the model include dbase data files and gridded inventories in ARC/INFO raster 'grids'. ARC/INFO software features allow the user to develop maps and other visualization tools.

MEASURE currently includes:

- *Modal (cruise, acceleration, deceleration, idle) activity*
- *Estimates of CO, HC, and NOx*
- *~100 emission specific technology groups*
- *Separate tree-based regression engine start and running exhaust emission rates*
- *Comparative MOBILE5a SCF running exhaust estimates*
- *Impacts of grade on acceleration*

MEASURE currently does not include:

- *Non-automobiles*
- *PM10 or PM2.5*
- *Evaporative emission estimates*
- *Vehicle deterioration effects*
- *Impacts of grade on engine load*

- *Impacts of accessory load*
- *Intersection specific estimates*

Included, but technically weak components:

- *Hourly, on-road traffic volume estimates*
- *Hourly, on-road traffic flow estimates*

Running and managing the model require technical skills in geographic information systems (ARC/INFO 7.0), transportation planning practice and software (TRANPLAN), air quality modeling (MOBILE 5a), C programming, and UNIX and MS-DOS operating systems. Substantial data collection and processing activity is required prior to model operation and is discussed in detail in the next section.

MEASURE is divided into 12 modules (see Figure 4.1). The modules and their associated input and output data files are managed by a single ASCII description file called 'Makefile'. The 'Makefile' is executed by a UNIX system utility called *make* (discussed in detail later). The modules roughly follow a tree structure in that one module may depend on the outputs of other modules. The modular structure allows individual components to be executed independently or in a fully connected process. The modules are grouped into five tiers: the spatial environment, the fleet characteristics, the vehicle activity, the facility emissions, and the emission inventory.

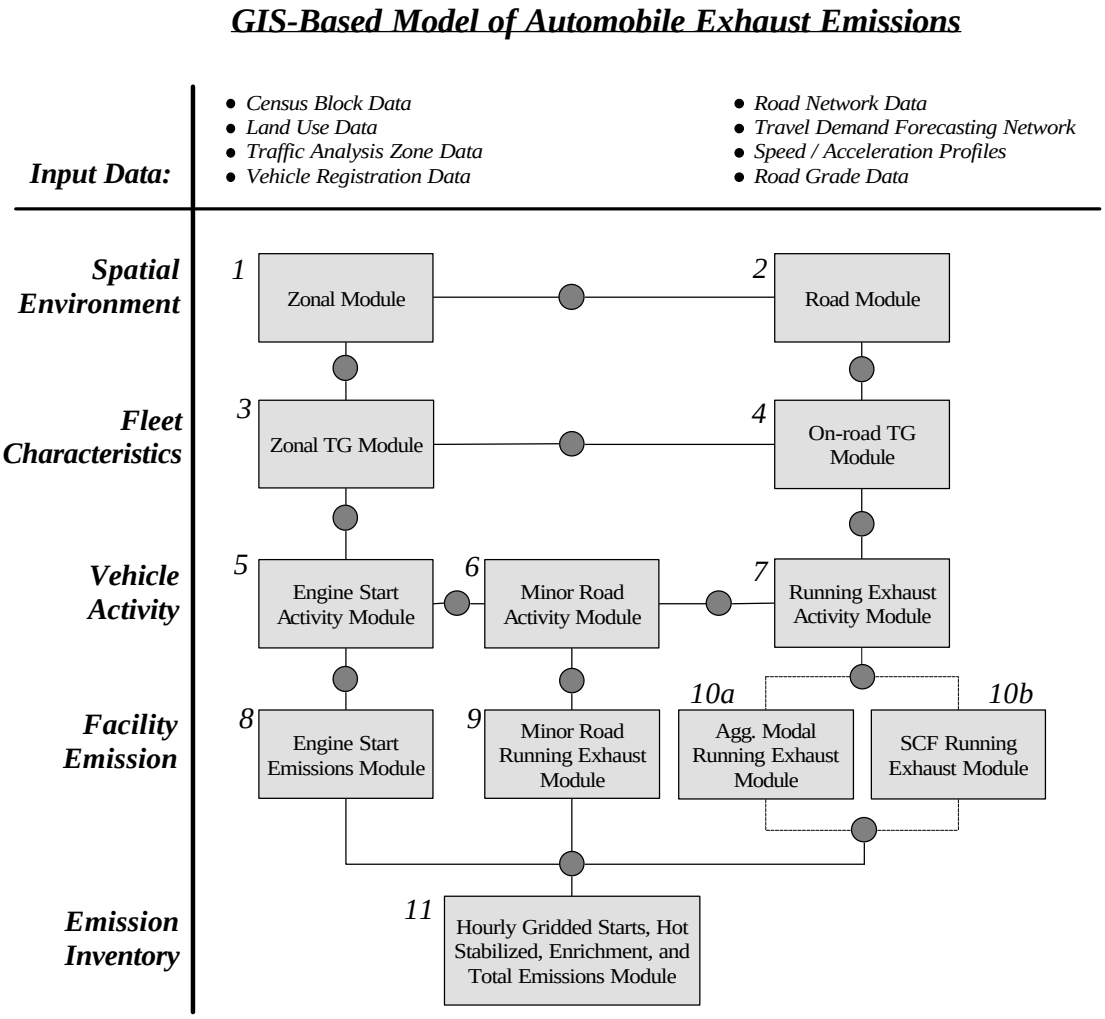

Figure 4.1 - Model Design

If all input files and associated software are in place, the modules are executed in the appropriate order by typing 'make emissions' at the UNIX command line. The test dataset (100 sq. km in northeast Atlanta) should take approximately four hours to complete on a Sun SPARCstation 10. A larger urban area may take as long as 24 to 36 hours to complete a full execution.

4.1. Input Files

This section will describe the datasets, fields, and directories needed to be resident in the system. A substantial amount of data collection and processing effort needs to be conducted prior to model operation. Several tools and guidelines are included to aid this process. As a general note, all ARC/INFO coverages and files containing coordinates should use the same geographic projection and use meters as the base unit.

4.1.1. Directory structure

The directory structure consists of one home directory and several data directories. The data directories are:

- zone*: stores all zonal data and coverages*
- road*: stores all lineal data and coverages*
- grid*: stores all vector grid data*
- em*: stores all emission estimate data*
- tg*: stores all technology group data*
- grade*: stores all road grade related data*
- raster*: stores all raster data*
- raw*: place to store backup copies of data and programs*
- aml*: stores all AML code*
- code*: stores all C code*
- templates*: stores a number of INFO file templates*
- sa*: stores all speed / acceleration profiles*
- modalmats*: stores all modal matrices*
- lookup*: stores all ASCII lookup files*
- temp*: stores temporary files used during program runs*

4.1.2. Zone.twt and zip.twt

Two ASCII files, zone.twt, and zip.twt, need to be created and stored in the *tg* directory. These files represent the area's vehicle characteristics. *Zone.twt* is a list of successfully addressed geocoded vehicles and an ID number representing the registered zonal location. *Zip.twt* are those vehicles that were not matched and therefore only the registered ZIP code location is included. The ASCII files have the same comma-delimited structure of:

- *zone or ZIPcode*
- *vehicle identification number (VIN)*
- *model year*
- *emission control code*
- *fuel delivery system code*
- *engine size (cubic inch displacement)*
- *vehicle dynamometer test weight*
- *CO high emitter flag*
- *HC high emitter flag*
- *NO high emitter flag*

Creating these files from an area's vehicle registration dataset is a lengthy, one-time process and therefore removed from the formal model. The first step in the process is to acquire the area's registration datafile that contains VIN, street address, and ZIP code. For a large urban area, this initial file can consist of millions of records (2.2 million for Atlanta). The file needs to be address geocoded (with offset) using tools available in ARC/INFO or other software, and an accurate, up-to-date, road database. Successful matches are stored in an ARC/INFO point coverage with the VIN. Unsuccessful matches are stored in an ASCII file of ZIP code and VIN. Keep track of the match rate, as it will be important later in the model. The point coverage is then overlaid with any zone coverage, preferably of census blocks, using ARC/INFO's *identity* command. The resulting zone-id and VIN are written to an ASCII file. The two ASCII files are processed through a series of programs written by individuals at Georgia Tech. These programs read these files, decode the VINs using vendor software, flag a random sample as high emitter for the three pollutants, and write the results to ASCII files called *zone.hne* and *zip.hne*. A separate program, *hne2twt.c*, reads these files and uses a lookup table to add vehicle dynamometer test weight.

4.1.3. Allroads (ARC/INFO coverage)

The *allroads* coverage is a line database of all roads in the area and should be as accurate as possible. It should be stored in the *road* directory. The database contains, in addition to standard ARC/INFO fields, the key fields called *arid* and *tfid*. These fields are identifier fields and are used to link datasets back to the individual road

segment locations. *Arid* is unique for each road segment. *Tfid* identifies the corresponding travel demand model link. Because travel demand models usually model major roads only, not all lines in *allroads* will have a non-zero *tfid*. Further, some travel model links will span a number of lines in *allroads* resulting in lines that have the same *tfid*. Establishing the *tfid* on the lines is completed through a manual process called conflation. The process of conflation involves selecting each travel model link, selecting all corresponding road segments, and joining the ids.

4.1.4. Tdfn.dat (INFO file)

The *tdfn.dat* INFO data file is a table stored in the road directory with the following items:

- tfid - *the key field to link to* allroads
- assign_gr - *the road classification code (1-interstates, 2-ramps, 3-major arteries, 4-minor arteries)*
- abvolume - *the oneway 24 hour volume*
- abspeed - *the oneway 24 hour average speed*
- abcap - *the oneway road capacity*
- bavolume - *the otherway 24 hour volume (blank if divided road)*
- baspeed - *the otherway 24 hour average speed*
- bacap - the otherway road capacity

This file is created by converting the area's travel demand forecasting model outputs into an INFO format. Programs by Georgia Tech are available that can complete this process for TRANPLAN models. If other travel modeling software is used, one may have to contact the vendor for a conversion package, or develop customized software that creates datafiles compatible with ARC/INFO's 'generate' and 'add from' commands.

4.1.5. Census (ARC/INFO coverage)

The *census* polygon coverage (stored in the *zone* directory) can be any zonal structure that contains a 'household' field. US Census blocks are preferred because they are available around the country and provide substantial detail with regard to population and household information. The *census* coverage contains the following fields:

- cbid - *the key field to link associated databases together*
- hu90 - *the household field*

Census data can be acquired from the US Census Bureau.

4.1.6. Land use (ARC/INFO coverage)

The *landuse* polygon coverage (stored in the *zone* directory) is a database of residential, non-residential, and commercial land uses. The coverage needs to contain only a single attribute field of *lu*. *Lu* is a character field that consists of 'RES' if the zone is residential, 'COM' if the zone is commercial, and 'NONRES' if the zone is non-residential. Areas without identifiers will be considered non-residential.

4.1.7. ZIP code (ARC/INFO coverage)

The *Zipcode* polygon coverage (stored in the *zone* directory) is a database of postal ZIP codes. The coverage needs only a single field called *Zipcode* and should be populated with current five-digit ZIP codes.

4.1.8. TAZ (ARC/INFO coverage)

The *TAZ* polygon coverage (stored in the *zone* directory) is a database of traffic analysis zones used in the travel demand forecasting process. The coverage needs a single key field called *tzid* and should be unique for every zone. Developing this datafile will probably require assistance from the local planning organization.

4.1.9. TAZ.dat (INFO file)

The *TAZ.dat* file stores all the trip generation information from the four step travel demand model. The fields are as follows:

- tzid - *the key field used to link the data to other zones*
- hbwprd - *home-based work productions*
- hbshprd - *home-based shopping productions*
- hbgsprd - *home-based grade school productions*
- hbuprd - *home-based university productions*
- nhbprd - *non-home-based productions*
- hbwatt - *home-based work attractions*
- hbshatt - *home-based shopping attractions*
- hbgsatt - *home-based grade school attractions*
- hbuatt - *home-based university attractions*
- nhbatt - *non-home-based attractions*

The productions and attractions fields listed above represent standard travel demand forecasting terminology for trip type. The fields should represent the 24-hour trip type quantities by TAZ. This file, like the *TAZ* coverage, needs to be converted to INFO from whatever format the planning organization provides.

4.1.10. Landmarks(ARC/INFO coverage)

The *landmarks* coverage is a point database of educational institutions. The points should have an identifier field that shows whether or not it is a university or grade school. The determination of which schools to include depends on the definition of the trip types from the travel demand forecasting model. They are used primarily to spatially allocate home-based-university trips and home-based-grade school trips.

4.1.11. Grid (ARC/INFO coverage)

The *grid* coverage will represent the spatial structure of the inventory estimates. Grid cells of any size can be used; however, they should not be smaller than the published accuracy of any of the input coverages. The coverage should contain one attribute called *gdid* .*Gdid* will be used as a key item to aggregate estimates from the zones and lines.

4.1.12. Grade.xy and grade.gr

The ASCII files *grade.xy* and *grade.gr* store road grade information collected from GPS units. The *grade.xy* file contains comma delimited fields of grade-id, x_coordinate, and y_coordinate. The coordinates must match the geographic projection system and units used by the coverages. The *grade.gr* file contains comma delimited fields of grade-id and grade. The files must be separate in order to read them into ARC/INFO as a point coverage. The model joins them in the *mr-act.aml* process.

4.1.13. Lookup files

Three lookup files are provided, *temporal.factors* (INFO file), *scf.csv* (ASCII file), and *twt.lu* (ASCII file). These files can be used in any model run, but may be updated. *Temporal.factors* provides the breakdown of vehicle volumes and trips by hour. *Scf.csv* provides MOBILE 5a speed-corrected emission factors by 5 mph speed increment and model year. It is used in developing the SCF running exhaust emission estimates. *Twtlu.asc* is a file of vehicle make, model, model year, and test weight. It is used outside MEASURE to add the test weight field to the vehicle characteristics files.

4.2. The Makefile

The *Makefile* is the most important file in the system to become familiar with. It manages the entire modeling process by checking file and code dependencies. The *Makefile* is interpreted by a system utility in UNIX operating system software (*make)*.

Make identifies file dates and times to determine when updates have been made, and, if needed, calls a series of actions. For example, the first dependency relationship in the *Makefile* is listed as follows:

```
$(em)/grid-em.dbf     : $(em)/scf-em.dbf \
                        $(em)/sz-em.dbf \
                        $(em)/mr-em.dbf \
                        $(em)/mz-em.dbf \
                        $(aml)/grid-em.aml
              /bin/rm -r $(raster)
              /bin/mkdir $(raster)
              /bin/rm -r $(temp)
              /bin/mkdir $(temp)
              $(arc) "&r $(aml)/grid-em.aml"
```

Make first checks to see if a file 'grid-em.dbf' exists, and if it does, it compares it with the last update of the other files. If one or more of the other files has a newer date than 'grid-em.dbf', the last five lines are executed. If 'grid-em.dbf' is current with respect to the other files, the code is not executed. The value of this is that it saves time in large complicated multi-program processes because only those portions that have been updated are executed.

The *Makefile* is segmented into twelve parts that represent the twelve modules listed in Figure 4.1. The twelve parts are called:

- *gridded_emissions*
- *scf_emissions*
- *eng_starts_emissions*
- *maj_rds_run_exh_emissions*
- *min_rds_run_exh_emissions*
- *min_rds_activity*
- *eng_starts_activity*
- *maj_rds_activity*
- *run_exh_tech_groups*
- *eng_starts_tech_groups*
- *rds_spatial_environment*
- *zonal_spatial_environment*

Any of the above parts can be executed by typing 'make <part>' and that particular module, and any non-up-to-date module it depends on, will be executed. By typing 'make gridded_emissions' all the parts are checked and executed if needed because the final gridded emission estimates depend on all of the other components.

In addition, two other parts have been added to aid in modeling. 'Make clean' will remove all data files except the original input files. This can be handy when one wishes to start with a clean slate. 'Make programs' will compile all the 'C' programs. This is handy when editing code and you want to make sure it works before running the Model.

Prior to running the Model on a new system, edits to the *Makefile* must be made. The first section entitled 'Variable Definitions' is as follows:

```
dir  = /proteus/home/wbachman/gismodel2
tg   = $(dir)/tg
zone = $(dir)/zone
road = $(dir)/road
lm   = $(dir)/landmarks
em   = $(dir)/em
grid = $(dir)/grid
c    = $(dir)/code
aml  = $(dir)/aml
raster = $(dir)/raster
temp = $(dir)/temp
arc  = /miranda/ceesri/arcexe70/programs/arc
cc   = /opt/SUNWspro/bin/cc
```

The variable 'dir' in the first line must be updated to the current path where the *Makefile* and other directories reside. Also, the 'arc' variable must be updated to the correct executable path for ARC/INFO. The same update must be made for the 'cc' variable, identifying the location of the C compiler.

4.3. The Modules

The modules, although separate, are linked together through dependent and co-dependent files and programs. Even though they are discussed separately, and function independently, their design and operation is affected by inputs and outputs of other modules.

4.3.1. Zonal Environment Module

The zonal environment module's purpose is to establish a polygon database called *sz* that will be used to develop engine start emission estimates. Figure 4.2 represents the entities involved in MEASURE. Figure 4.3 represents the flow of the code. The polygon database is created by spatially joining four input coverages, *census, taz, landuse,* and *ZIPcode*. The process of joining the databases involves the use of the ARC/INFO command 'identity'. The input coverages may have relative

inaccuracies that can cause 'sliver' polygons to be created. Operationally, this is not important, but it does affect relative accuracy. A line on one coverage may represent the same feature as the line on another coverage; however, the spatial representations of the line may differ. To lessen the impact of this problem, a 'fuzzy' tolerance is designated. Any lines that fall within the tolerance distance will be considered the same feature. The fuzzy tolerance should be set to the size of the worst reported absolute accuracy. In most instances, this will be the Census Bureau files that maintain an accuracy of 30-100 meters.

The module also creates a data file that maintains various attributes accumulated through the integration of the four input databases. These fields are land use (RES, NON-RES, and COM), housing unit per square kilometer, and all the TAZ trip generation estimates. The housing unit rate is multiplied by the *sz* area to predict the number of households. The amount of error introduced by disaggregating household data is a function of the size and accuracy of the original household database. If the Census blocks are used, a common source of large scale household data, the disaggregation errors will be small because the new zones will be similar to the blocks.

4.3.2. Road Environment Module

The purpose of the road environment module is to create the spatial structure used to represent running exhaust emissions. The road environment module divides the *allroads* coverage into roads modeled by the area's travel demand forecasting model, and those that are not modeled. The modeled road segments are used to create the *mr* (major road) line coverage. The *mr* lines form the boundaries for *mz* (minor zone) polygons that represent a zonal aggregation of minor roads. Figures 4.4 and 4.5 represent the entities and program code flow, respectively.

The module does not result in any loss of accuracy. Some minor zone polygons become insignificant due to the techniques used for generation. Median areas bounded by two parallel roads become minor zones. They bias the network travel distances slightly (determined later).

4.3.3. Zonal Technology Groups Module

The purpose of the zonal technology group module is to convert the outputs of the vehicle characteristics process into zonal technology groups. The vehicle characteristics process and zonal TG modules are displayed in Figures 4.6 and 4.7. Figures 4.8 and 4.9 describe the pre-processing steps. The module reads the two ASCII input files of *zone.twt* and *zip.twt* and creates four dbase files; *sztg.dbf, retg.dbf, scftg.dbf,* and *regiontg.dbf.* The module contains four programs; *techgr.c, regiontg.c,*

jointg.c, and *sz-tg.aml*. *Techgr.c* (Figure 4.10) is executed first and it assigns each vehicle into a technology group for each pollutant and summarizes by zone or ZIP code. The resulting distributions are written to a series of ASCII files. The *regiontg.c* program is similar in structure to *techgr.c,* but it reads all the vehicles and determines a regional technology group distribution. The *jointg.c* (Figure 4.11) program brings the separate zonal and ZIP code distributions using a control file (*szzp.asc)* containing zonal ids and corresponding ZIP codes. It is important to adjust the code to the address match success rate developed in the preprocessing. Currently the rate is set to 81% zonal and 19% ZIP code, suggesting that 81% of the vehicles were successfully geocoded (the match rate for the test dataset). *Jointg.c* writes three ASCII files, one for engine starts, one for running exhaust, and one for the SCF approach. *Sz-tg.aml* reads the outputs of the C programs and converts the files to dbase files.

Few errors are generated with this module; however, many errors occur in the pre-processing data development stage. These and other errors are discussed in Chapter 4. As much as 40% of vehicle representation is lost due to decoding errors. Errors resulting from the address matching process further degrade the spatial quality; however, the use of distributions of groups may lessen the impact.

4.3.4. Major Road Technology Groups Module

The purpose of the major road technology group module is to estimate on-road fleet distributions. The major road technology groups module reads the zone-based *retg.dbf, regiontg.dbf* and *scftg.dbf* (Figure 4.12 shows the module). By utilizing the *sz* and *mr* coverages, it develops road segment specific fleet distribution estimates. This is accomplished in the *mr-tg.aml* process by determining a 'local' fleet distribution and combining that with a regional fleet distribution (Figure 4.13). The local fleet is defined as an aggregation of vehicles registered within a 3 km radius of the road segment. This process takes the longest of all of the modules because it has to develop the combined distribution for each of multiple thousands of road segments individually. The output of this process is a single dbase file called *mrtg.dbf* containing the road segment id and the technology group percentages.

Significant errors can occur in this process due to the unvalidated nature of the assumptions. The assumption that the on-road fleet can be predicted by combining distributions from a region and a 'user-defined' local fleet is loosely based on research completed by Tomeh, 1996. Some evidence described in Chapter 6 is available that defends the estimate.

4.3.5. Engine Start Activity Module

The engine start activity module predicts the number of engine starts by *sz* poly and by hour of the day. Figures 4.14 and 4.15 represent the flow of the engine start activity estimation process. This is accomplished by reading the TAZ trip generation results file (*taz.dat*) and disaggregating the trips to *sz* zones using landuse, housing units, zone size, and school / university landmarks. All home-based trip origins are assigned to residential land uses based on household density. Shopping return trip origins are allocated to commercial land uses based on area. University return trip origins are allocated to zones that contain university landmarks based on enrollment. Grade school return trip origins are allocated to zones that contain grade school landmarks based on enrollment. Work return trip origins are allocated to non-residential zones based on area. If conditions exist that do not allow allocation of known trips using the above rules, trips are allocated to all zones based on area. Once trips are allocated to *sz* polys, they are disaggregated to hours of the day using hourly factors developed from Atlanta surveys for each trip purpose. The output of the module is a dbase file of total trip origins by zone and time of day.

Error in this process is largely the result of poor input data quality. The use of the travel demand forecasting values defines the highly aggregate original form of the data. A major missing component is the inclusion of intra-zonal and external travel patterns. Engine starts that result from individuals that go to destinations within the TAZ will not be accounted for. Intra-zonal / external trips that originate outside the study area and stop inside the site, will also not be represented.

4.3.6. Major Road Running Exhaust Activity Module

The running exhaust activity module determines hourly traffic conditions on all major road segments found in the *mr* line coverage. Figures 4.16 and 4.17 illustrate the process. Predicted conditions from the travel demand forecasting model output file (*tdfn.dat*) are combined with an hourly factor to predict road segment specific hourly volume, average speed, and LOS. The AML process also reads the grade ASCII files, joins them into a point coverage, assigns each point to the closest major road segment, and summarizes the grade points into five intervals. The data summaries of grade, static conditions, and dynamic conditions are written to a dbase file called *mr-act.dbf* storing results by the *arid* key field.

Besides input data error, the incorporation of road grade causes some significant problems. While the anticipated technique for collecting road grade (GPS) results in absolute positional accuracy that exceeds the road database, it is the relational accuracy that allows useful locational data to be collected. The process of snapping points to the lines in order to develop the relational structure can produce poor results

around intersections and close parallel roads. Further analysis on this subject is available in Chapter 6.

4.3.7. Minor Road Activity Module

The minor roads activity module develops estimates of the mean travel time and hourly trips occurring within each of the minor zones. Figures 4.18 and 4.19 illustrate the process. The shortest path is determined between the *sz* polygon centroids and the closest node of the major road network. The aggregate travel time is developed along with summaries of the hourly trip production at the zone (from the *sz-act.dbf* file). The final output is a dbase file called *mz-act.dbf* containing an aggregate travel time value and hourly trip production for each minor zone.

The potential errors in estimating the travel time are substantial. However, the alternative use of centroid connectors or zonal surface area would not allow a measure of network configuration within the zone.

4.3.8. Engine Start Emissions Module

The engine start emissions module predicts hourly CO, HC and NOx emissions for engine starts in *sz* polygons. Figures 4.20 and 4.21 illustrate the process flow. Generally, engine start technology groups are combined with emission rates and estimates of the number of hourly trips to develop the emission estimates. Output is written to a dbase file and stored in the *em* directory.

Emissions are elevated at the start because catalyst control equipment needs to operate at high temperatures. While the actual emissions are dispersed as the vehicle travels for the first few minutes, the model allocates the entire start portion to the origin *sz*. Therefore, spikes of high emissions may be estimated at high population density locations when in actuality the emission production at that location would be lower. This source of error in position is significant. Although the 'puff' allocation is significant in identifying the original sources of high start emissions, the actual location that the emissions entered the atmosphere is misrepresented.

4.3.9. Minor Road Running Exhaust Emissions

The minor road running exhaust module has the task of predicting hourly emissions of CO, HC, and NOx, given the activity conditions provided by *mz-act.dbf*. Figures 4.22 and 4.23 show the module flow. While information regarding technology groups is available for minor zones, traffic flow and modal conditions are not. Speed and acceleration data have not been collected to provide clues for determining local road profiles. For this reason, the module uses an average high and normal emitter

running exhaust emission rate for the three pollutants of concern. As soon as data become available, the minor zone emissions can follow a similar track as the major road emissions model.

4.3.10. Major Road Running Exhaust Emissions

The major road running exhaust emission's module calculates an aggregate hot-stabilized and enrichment emission estimate for all major roads. The module is represented in Figures 4.24 and 4.25. Estimates of vehicle activity and on-road technology groups are combined with technology and operating mode specific emission rates to develop hourly, road segment estimates of CO, HC, and NOx. Matrices of modal variables are available as lookup files. These files match the format of the speed / acceleration profiles, and consist of zeros and ones. The two files are multiplied together, and the resulting speed and acceleration file bins are summed. The values represent the fraction of activity in that particular mode.

4.3.11. Gridded Emissions

The gridded emissions module develops the total emission estimates by grid cell in a raster format. It has the task of overlaying a user-defined grid polygon coverage with the *sz*, *mr*, and *mz* coverages, and aggregating all the emission estimates (weighted by the area or length). All of the emission mode estimates (engine start, minor zone, major road, and SCF) are converted to raster datasets. The engine start, minor zone running exhaust, and major road running exhaust are summed together to develop estimates of total hourly CO, HC, and NOx. The process for obtaining gridded emissions is shown in Figures 4.26 and 4.27.

4.4. Conclusion

Chapter 4 described how the applied model design from Chapter 3 was translated into a functional computer model. Detailed descriptions of the input files and system structure are provided. Flow charts for the various modules and programs are provided as well. The model includes both SCF emission rates and modal emission rates. Engine starts and running exhaust are calculated separately for normal and high emitting vehicles. Gridded, hourly emission estimates are produced.

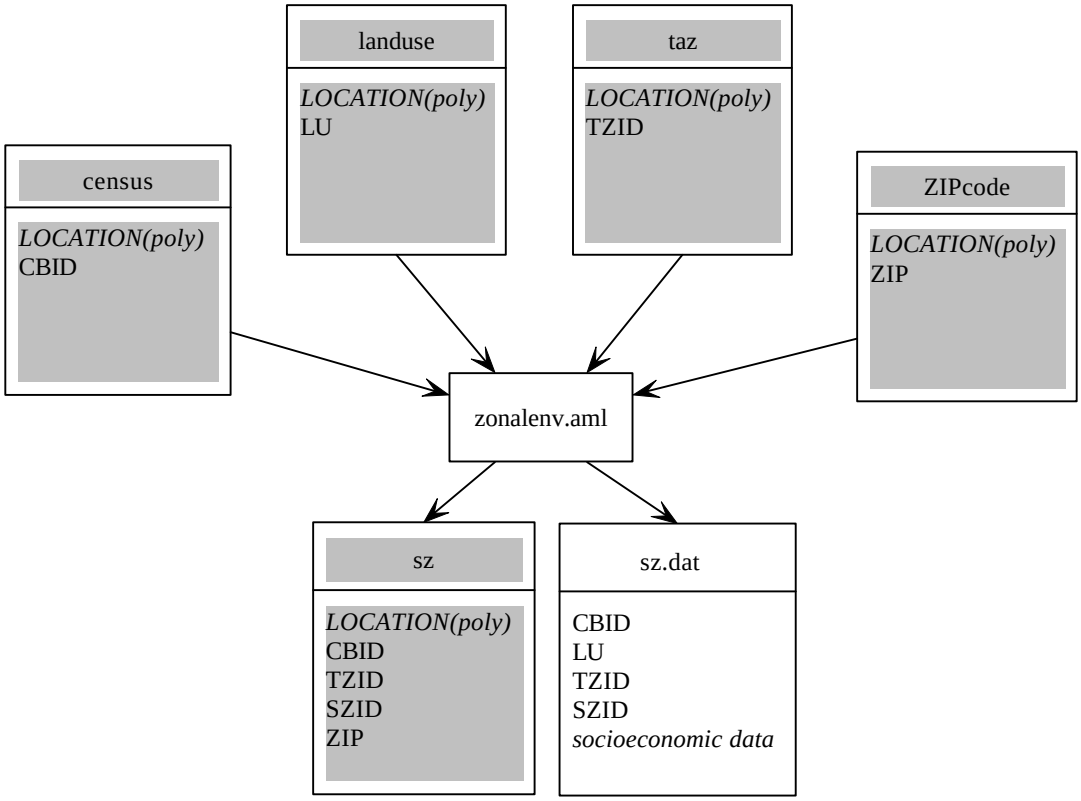

Figure 4.2 - Zonal Environment Entities

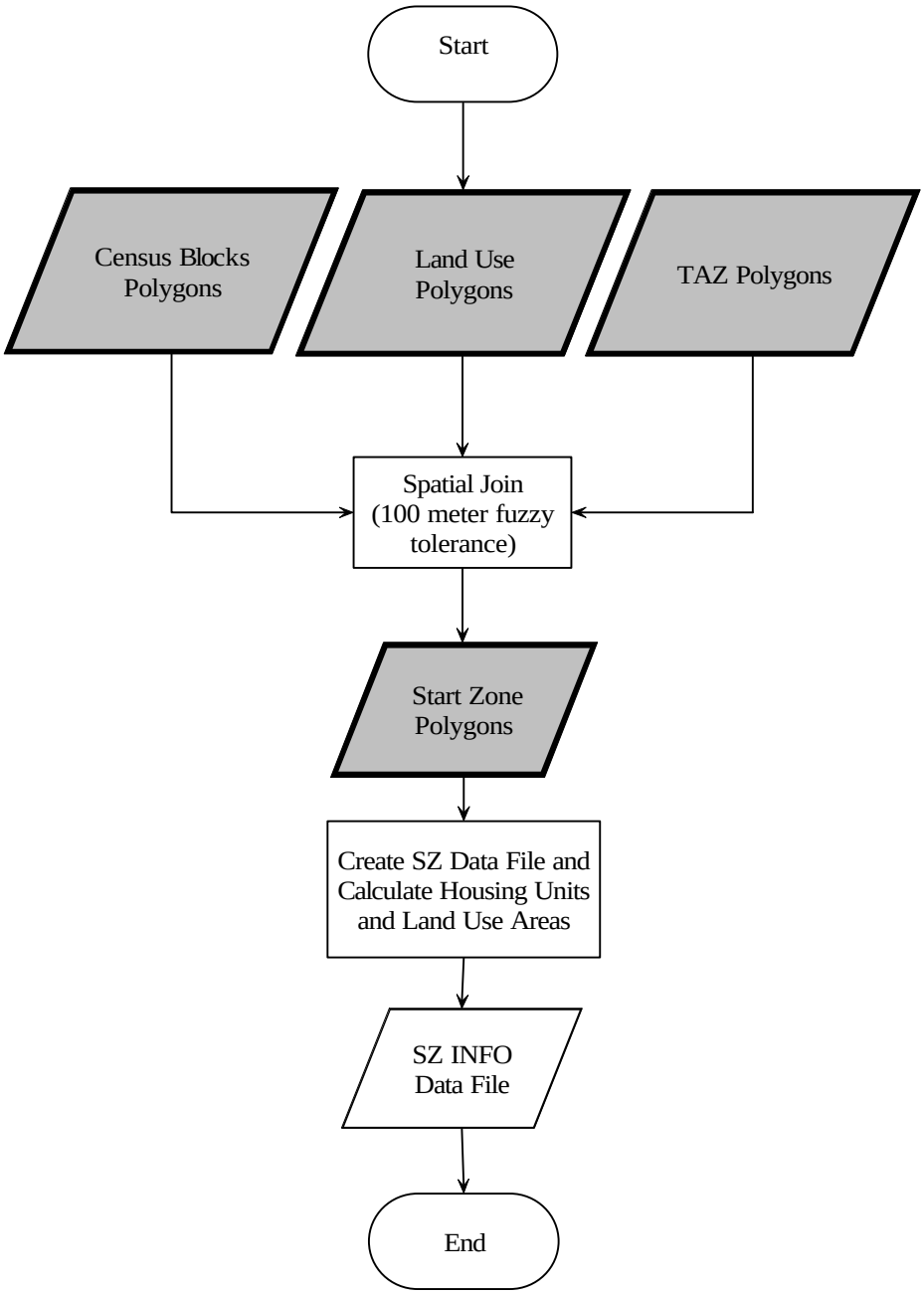

Figure 4.3 - *Zonalenv.aml* Flow Chart

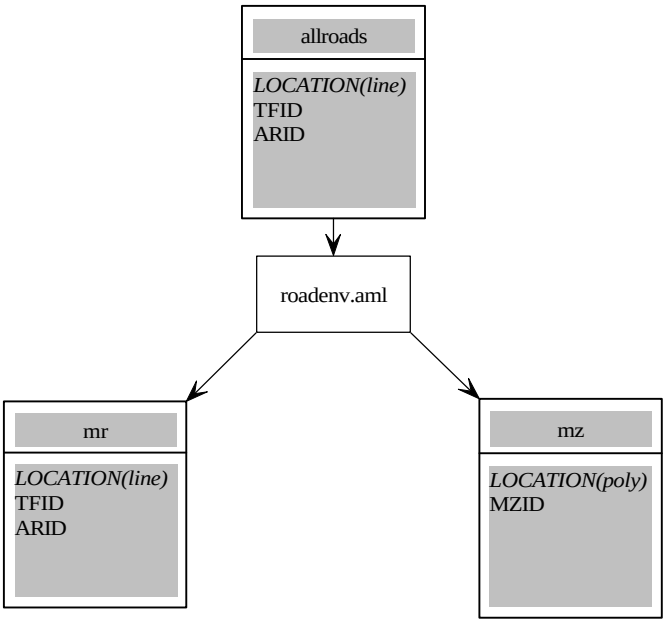

Figure 4.4 - Road Environment Entities

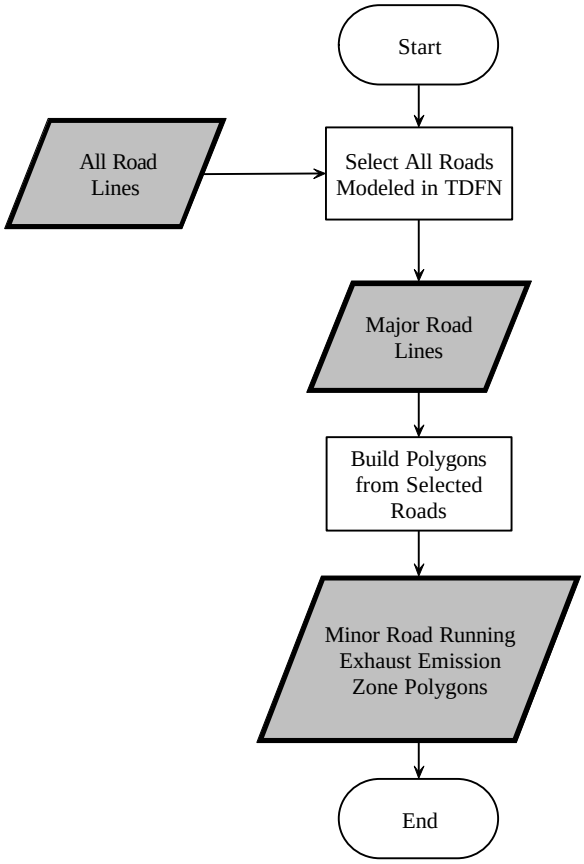

Figure 4.5 - *Roadenv.aml* Flow Chart

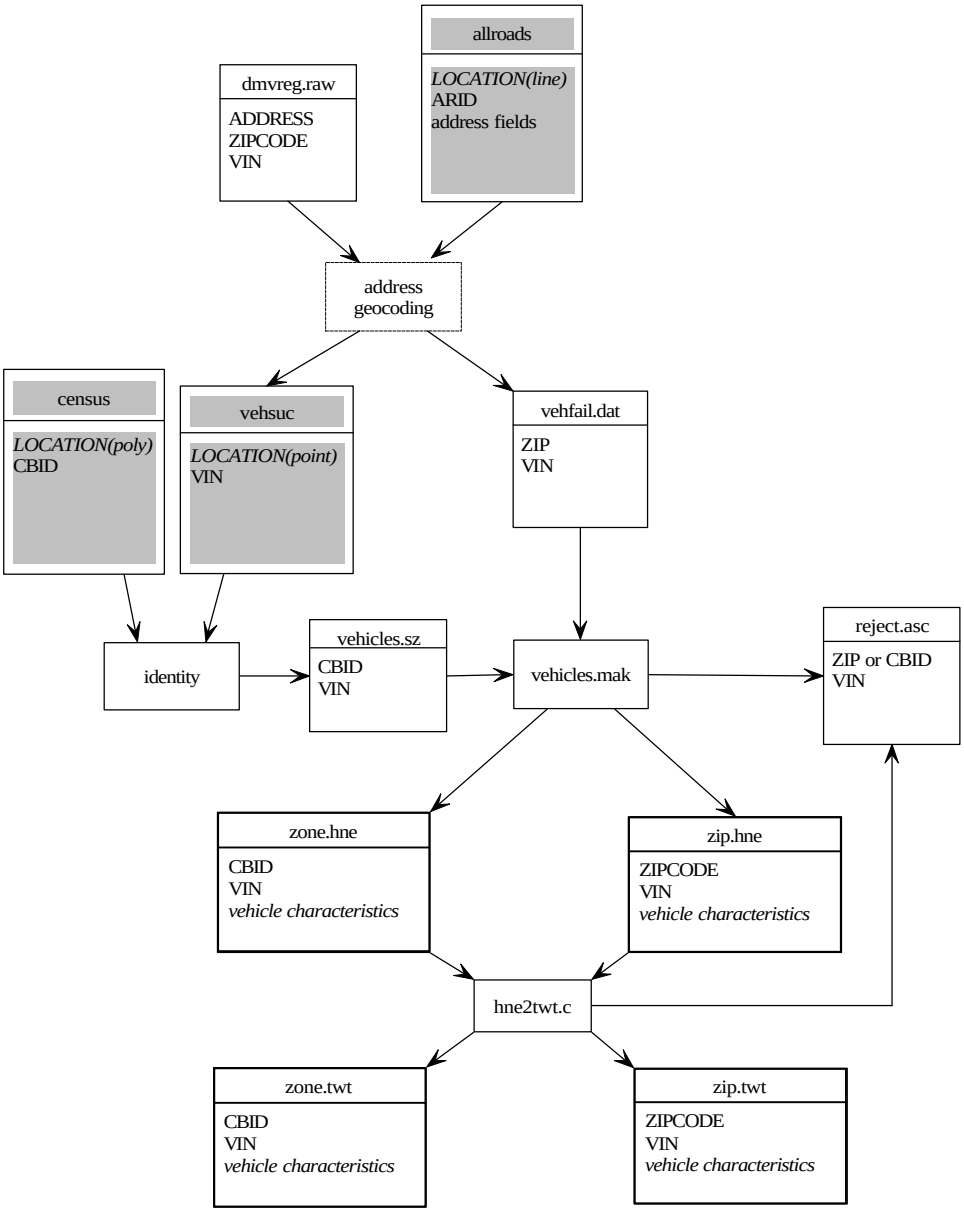

Figure 4.6 - Vehicle Characteristic Entities

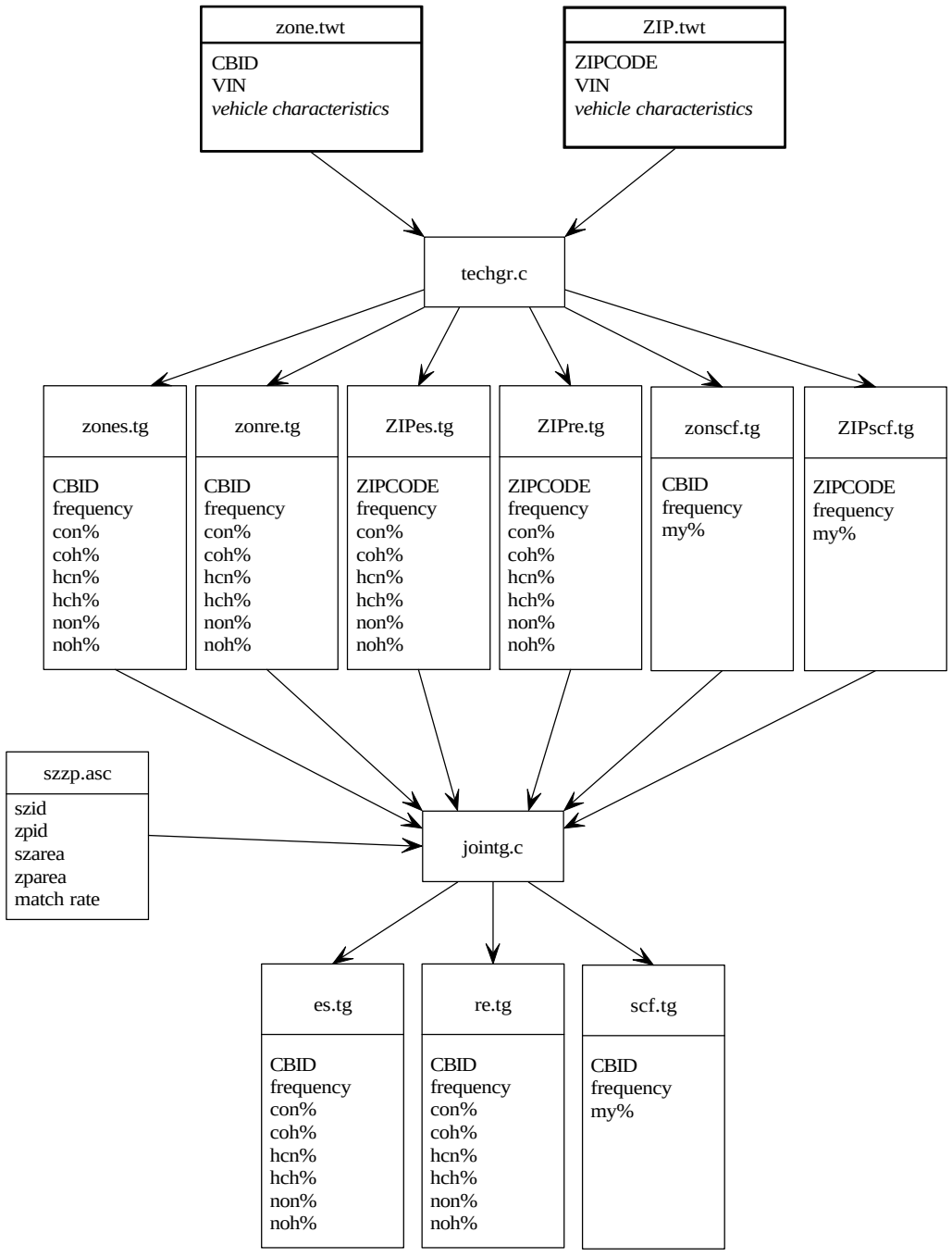

Figure 4. 7 - Technology Group Entities

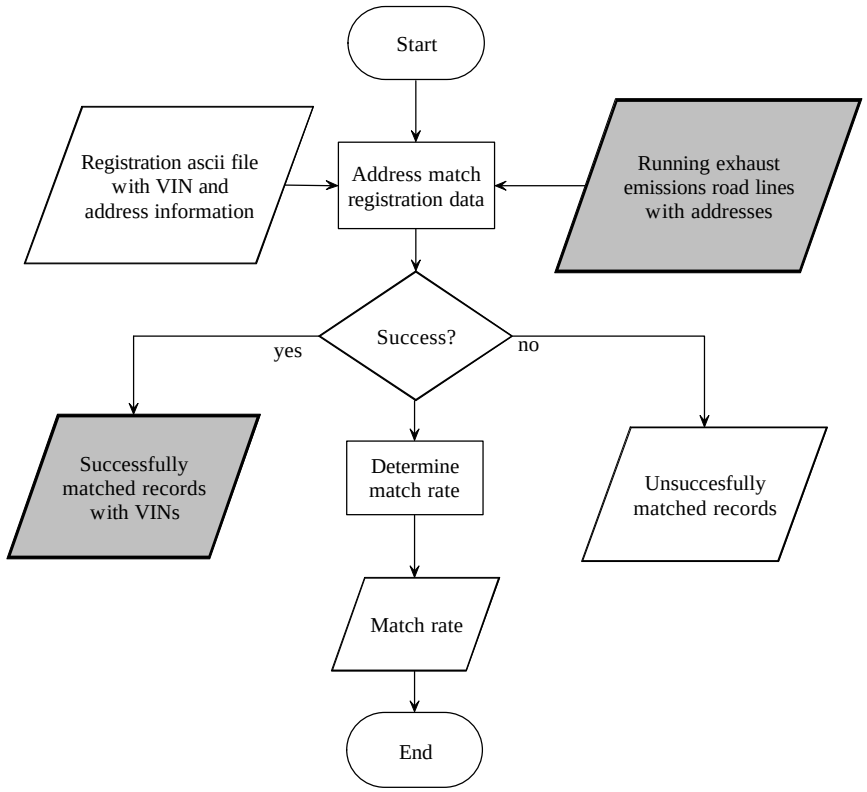

Figure 4. 8 - Address Matching Flow Chart

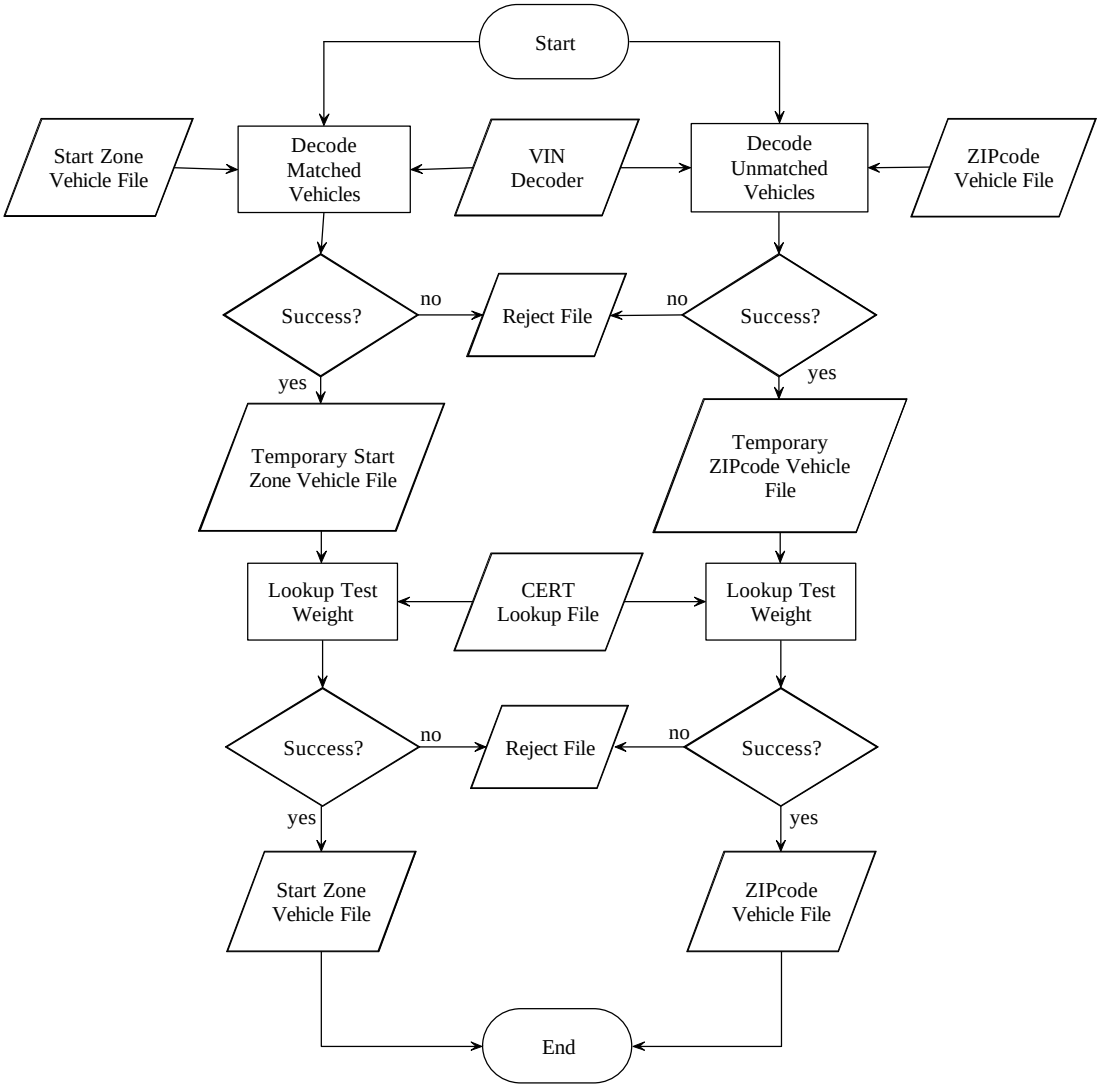

Figure 4. 9 - *Vehicles.mak* Flow Chart

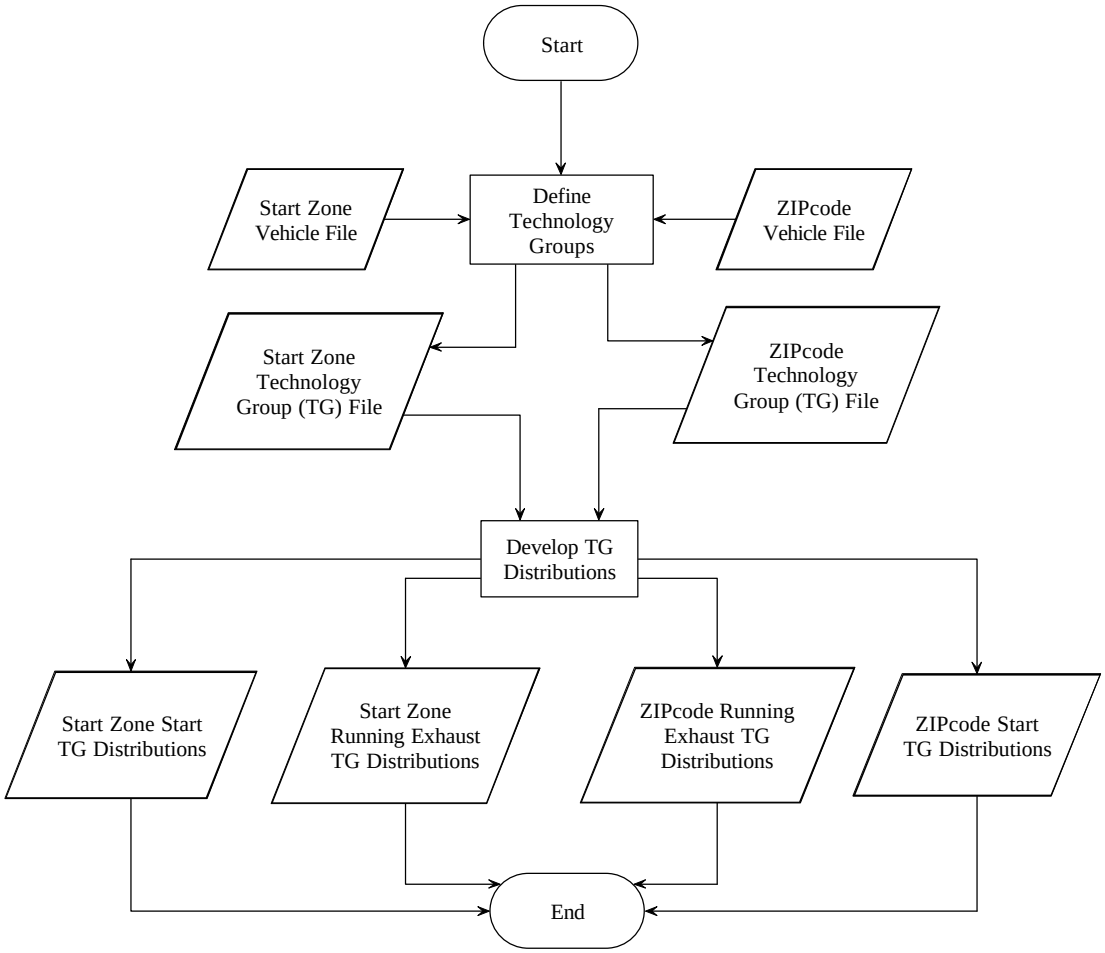

Figure 4. 10 - *Techgr.c* Flow Chart

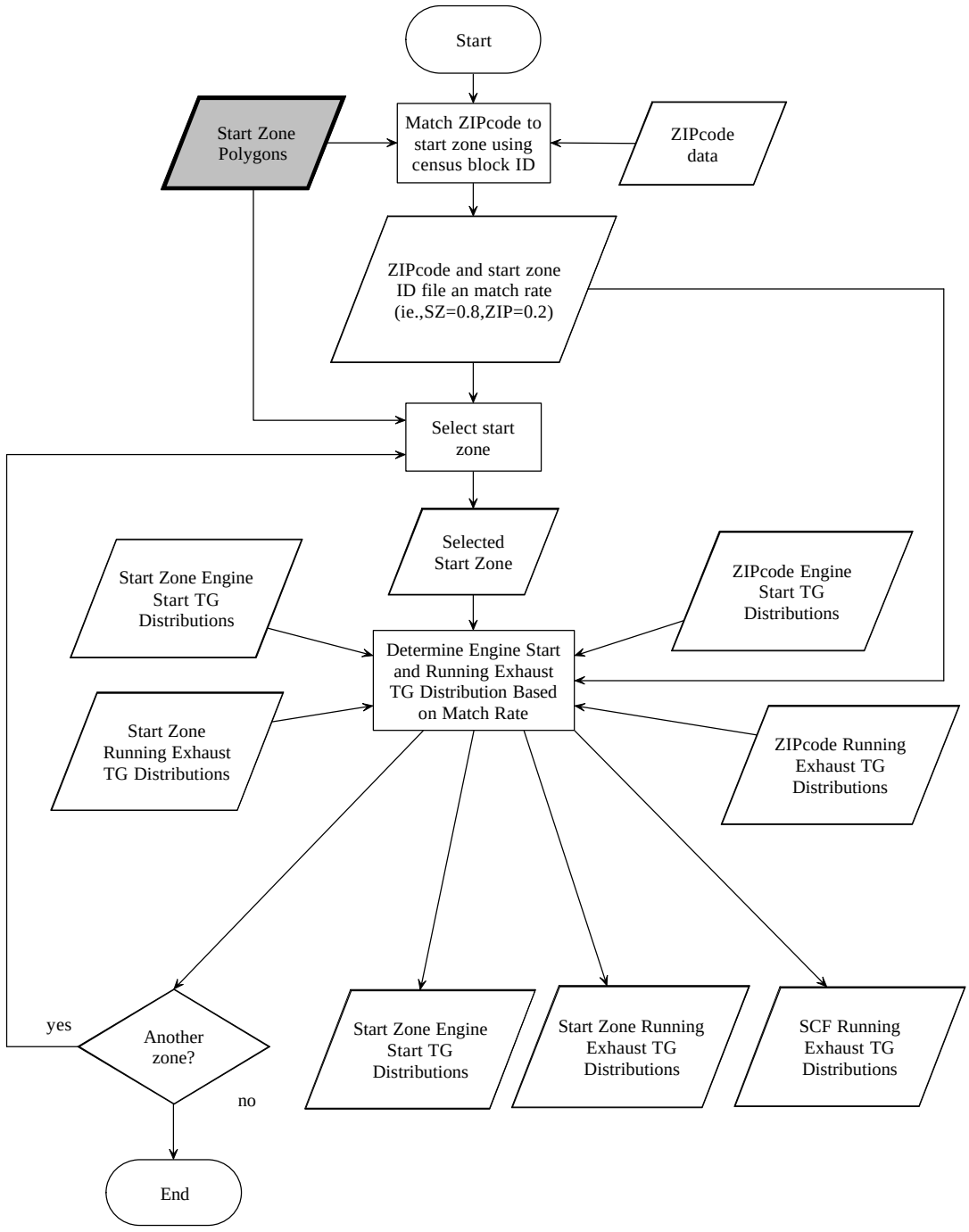

Figure 4.11 - *Jointg.c* Flow Chart

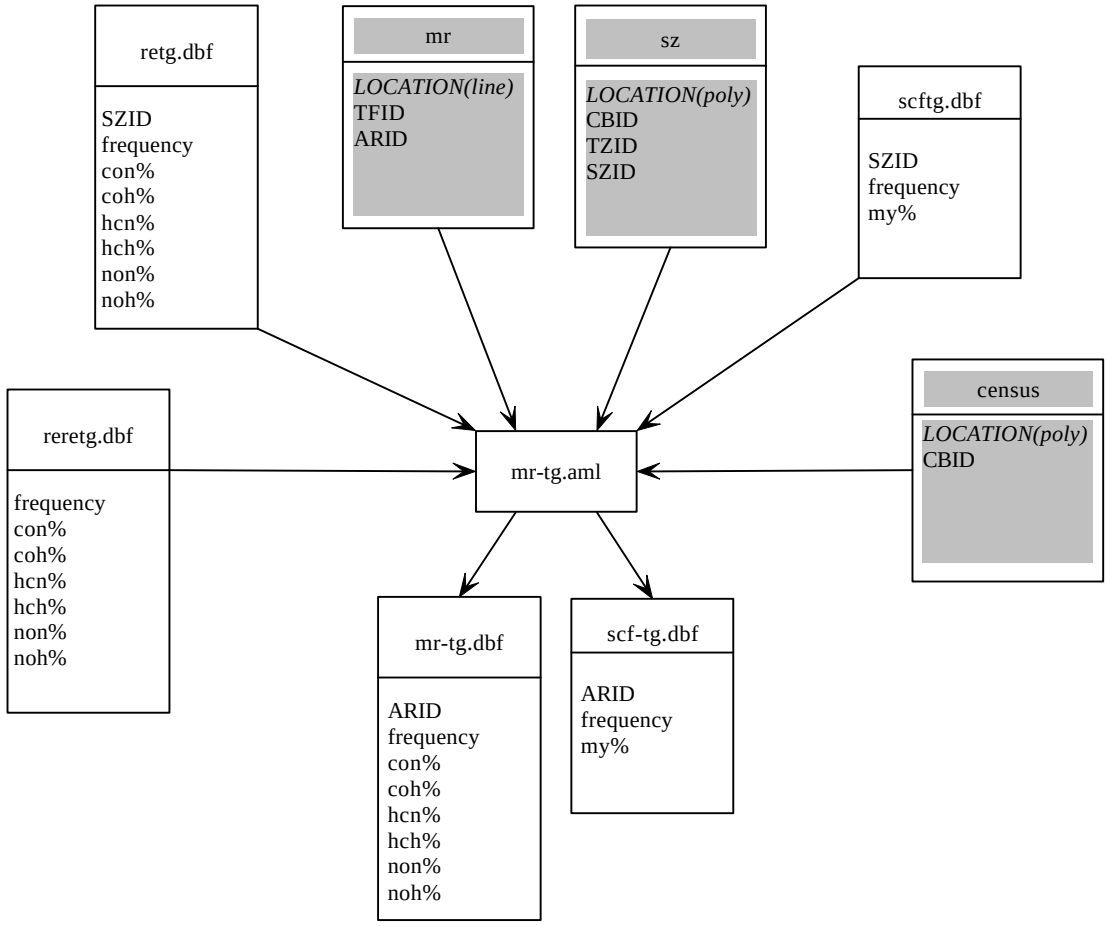

Figure 4.12 - On-Road Technology Group Entities

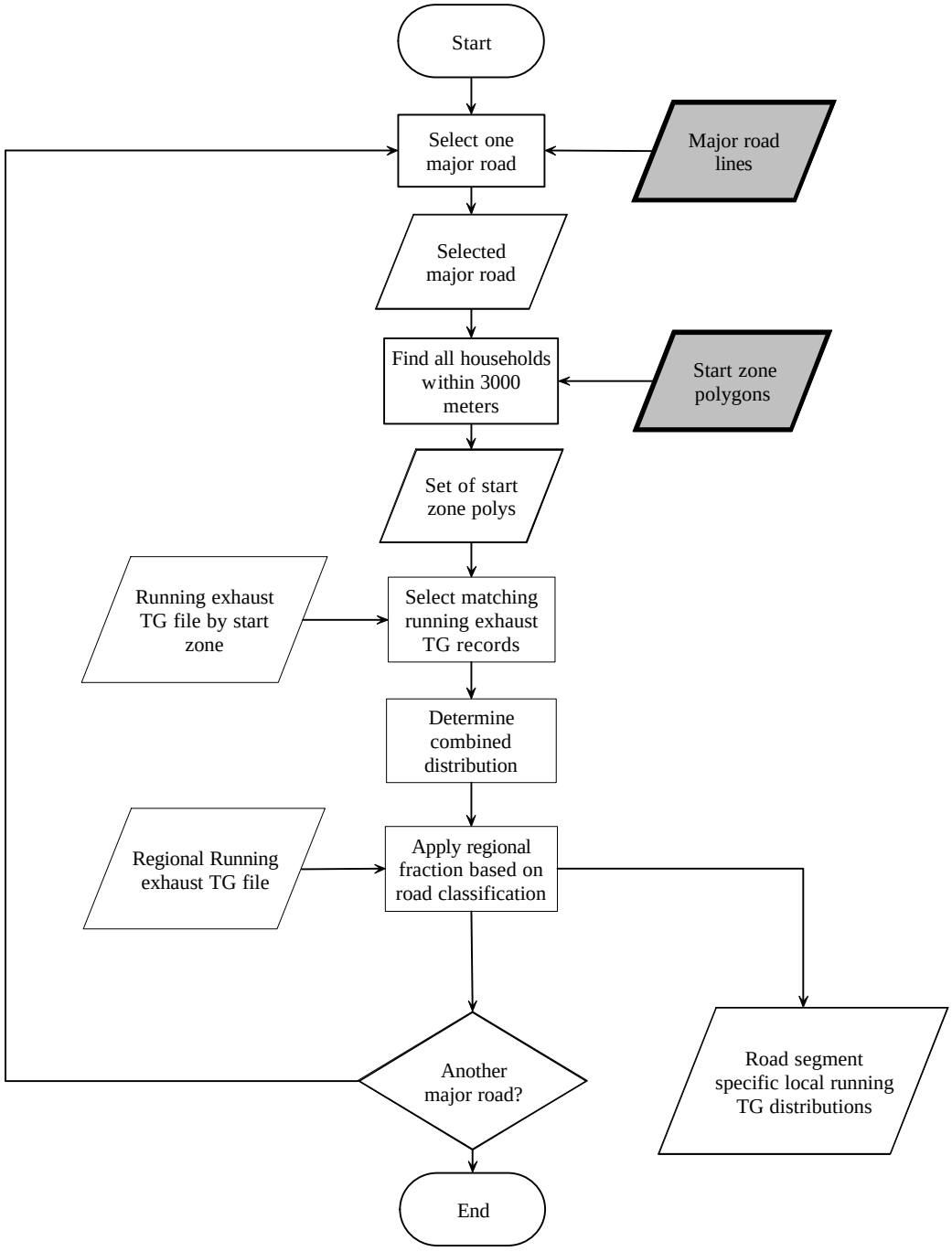

Figure 4.13 - *Mr-tg.aml* Flow Chart

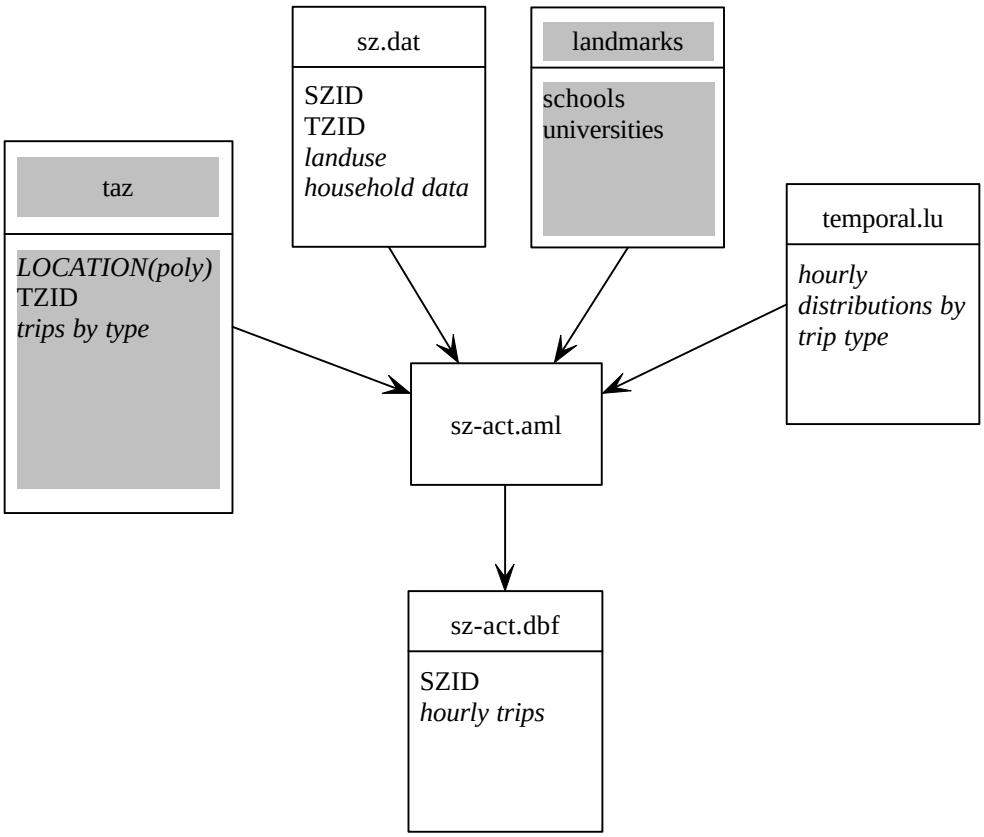

Figure 4.14 - Start Zone Activity Entities

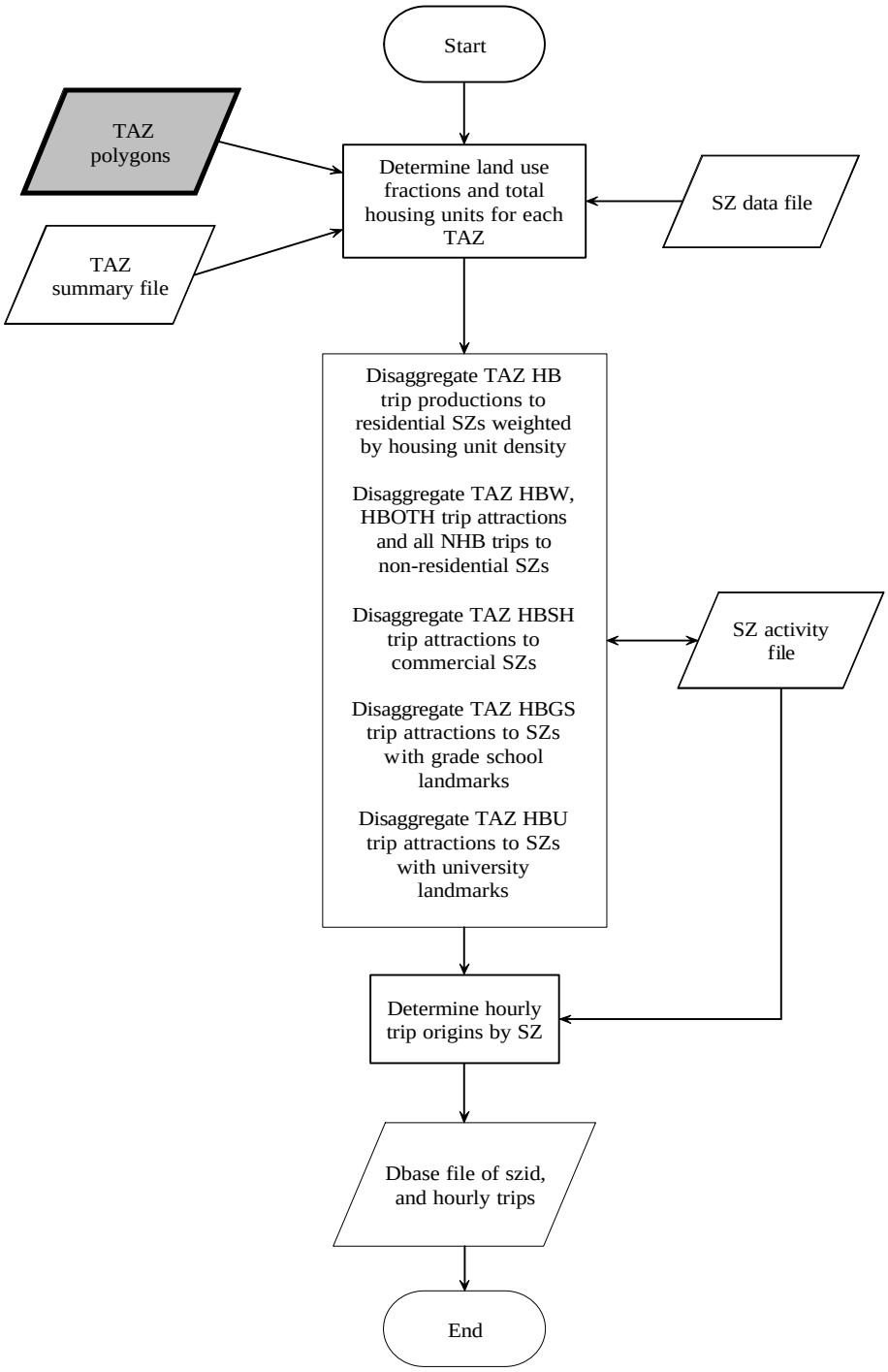

Figure 4.15 - *Sz-act.aml* Flow Chart

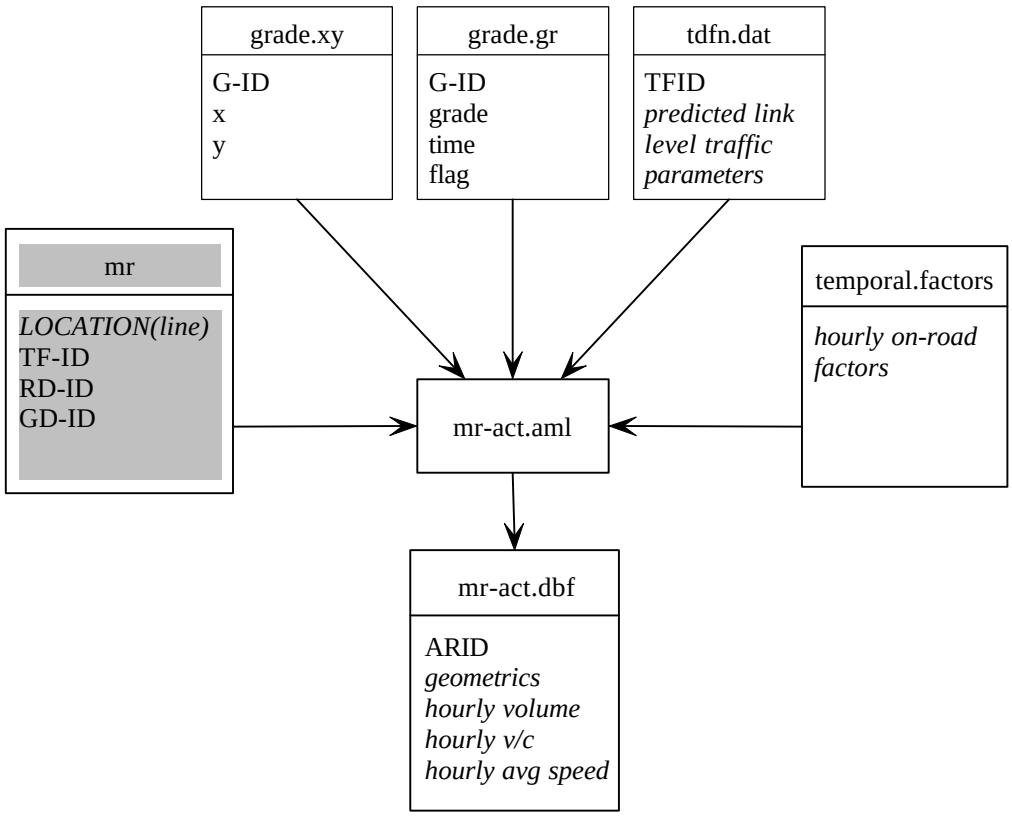

Figure 4.16 - Major Road Activity Entities

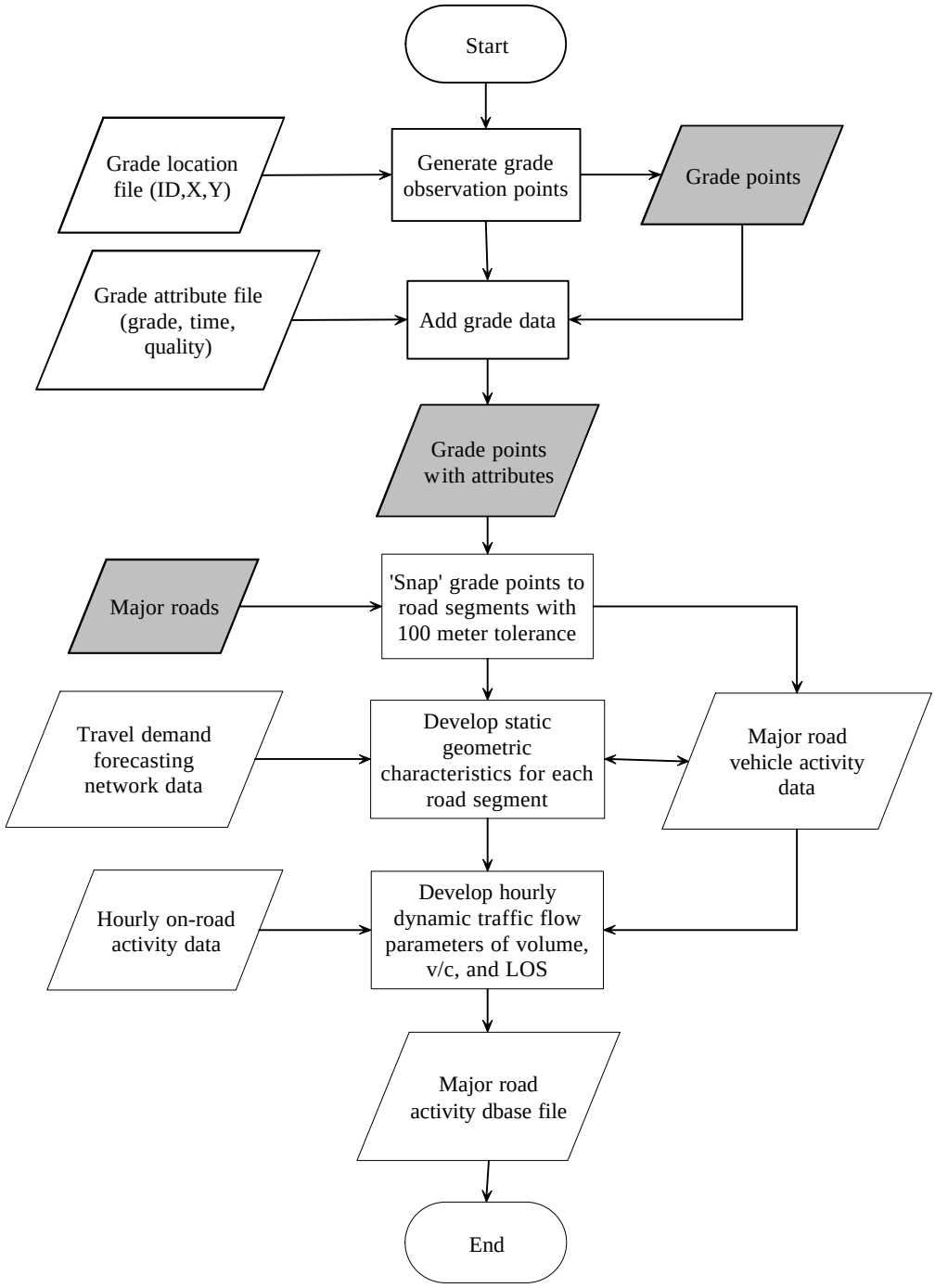

Figure 4.17 - *Mr-act.aml* Flow Chart

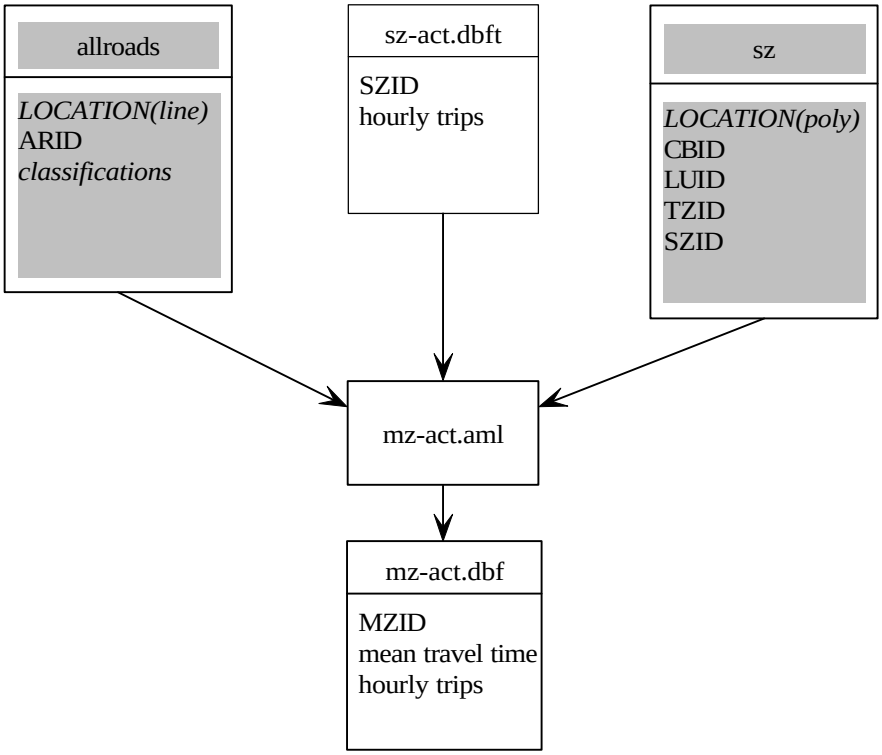

Figure 4.18 - Minor Road Activity Entities

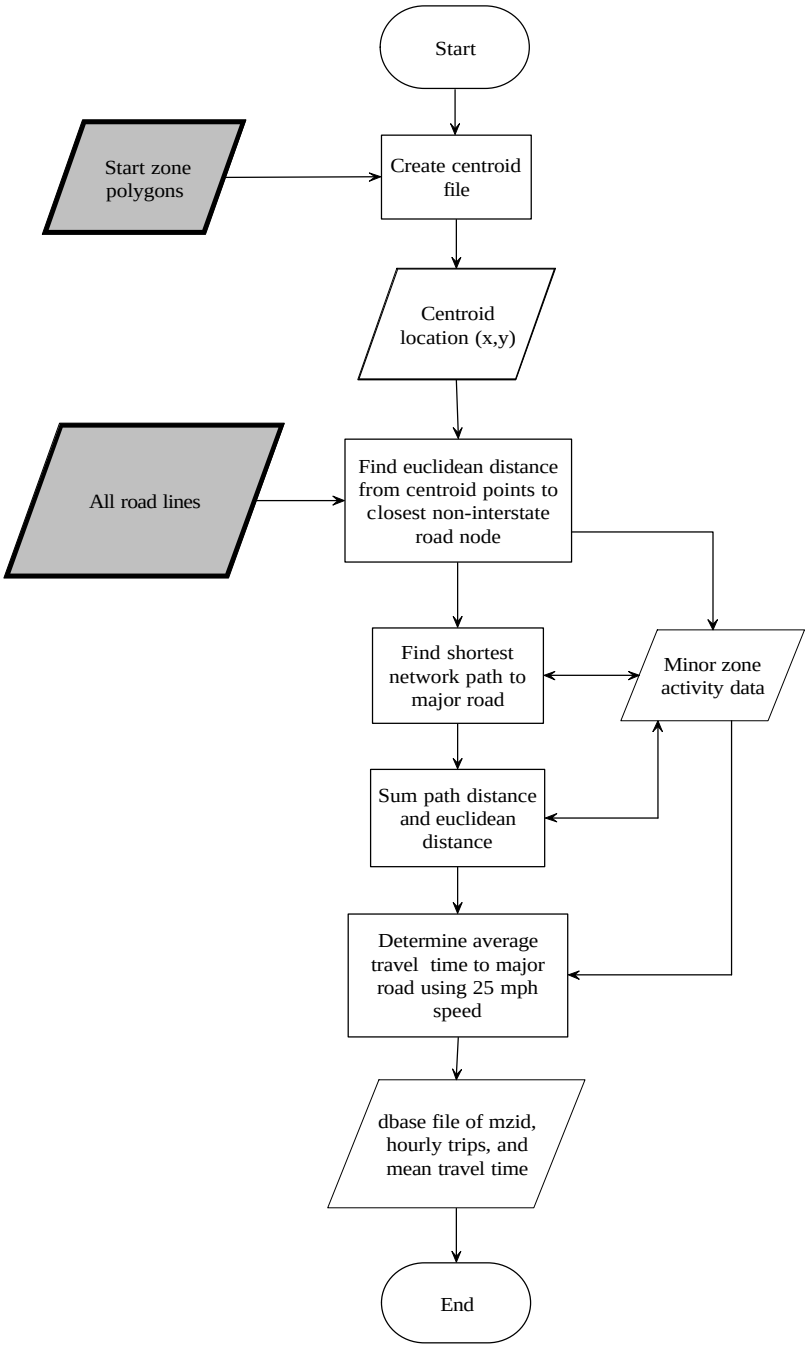

Figure 4.19 - *Mz-act.aml* Flow Chart

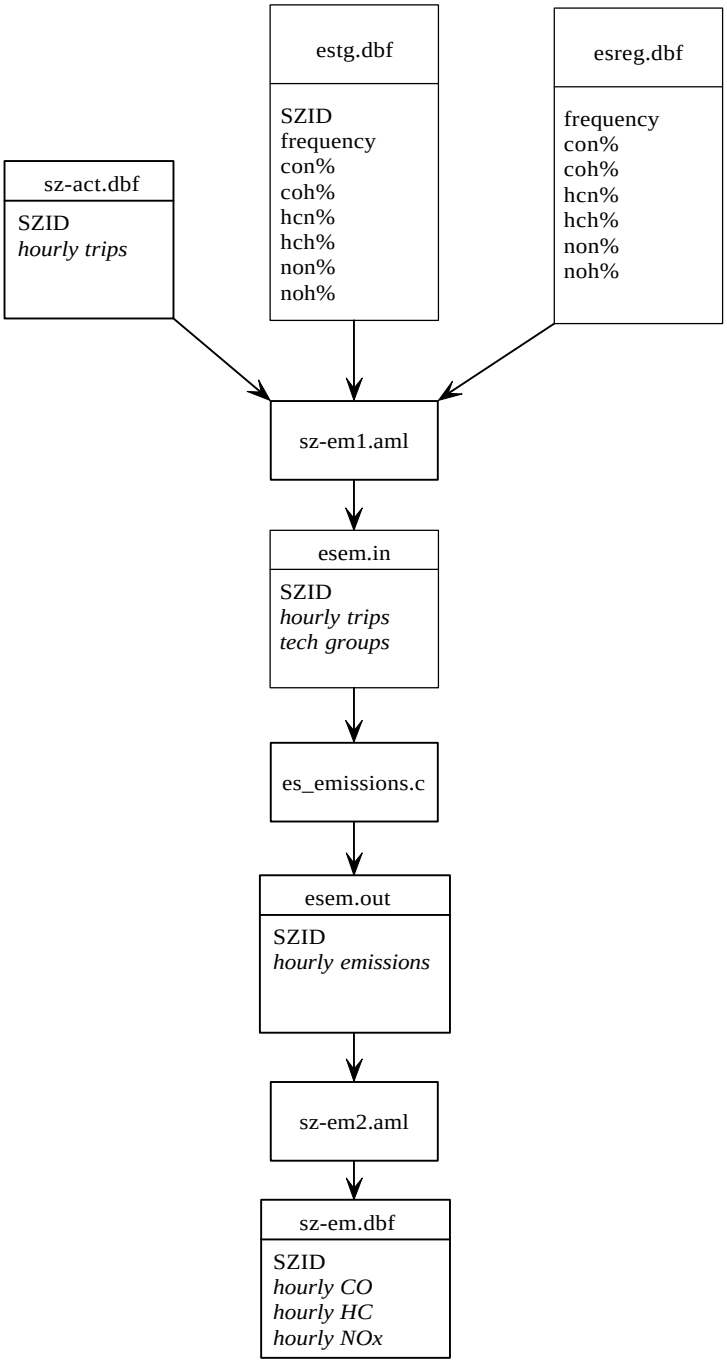

Figure 4.20 - Start Zone Emissions Entities

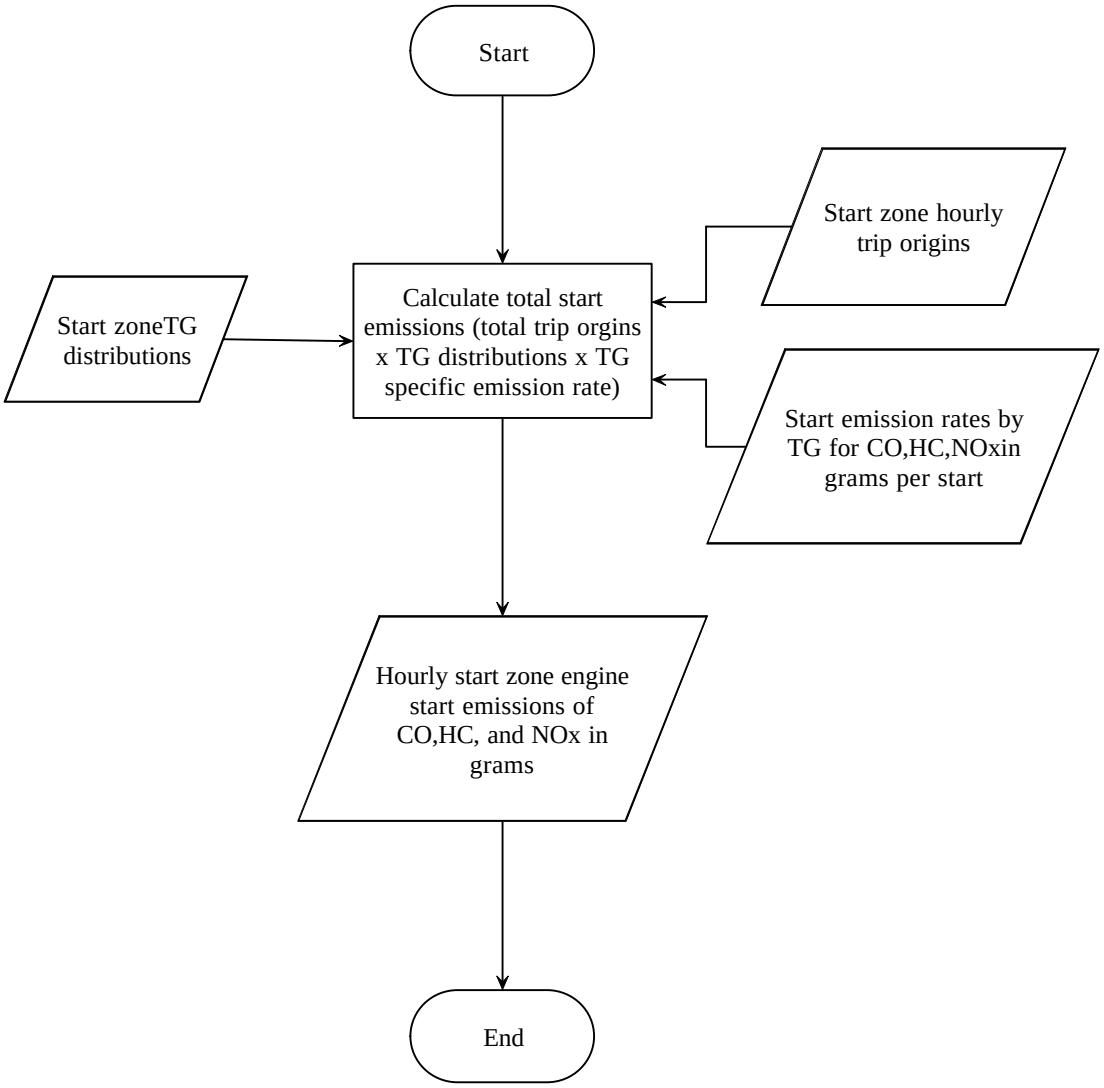

Figure 4.21 - *Es_emission.c* Flow Chart

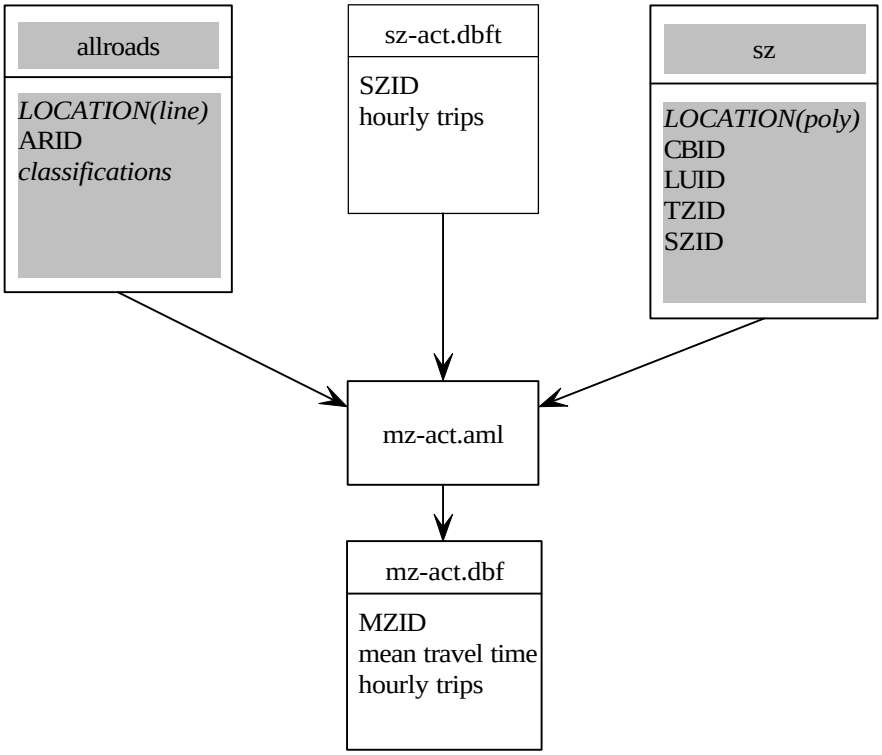

Figure 4.22 - Minor Zone Activity Entities

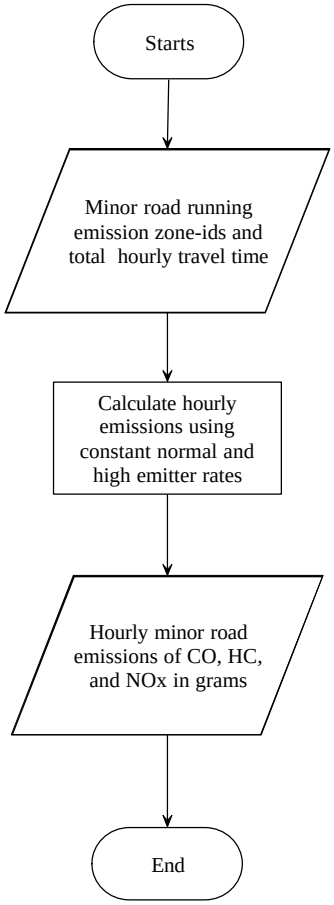

Figure 4.23 - *Mz-em.aml* Flow Chart

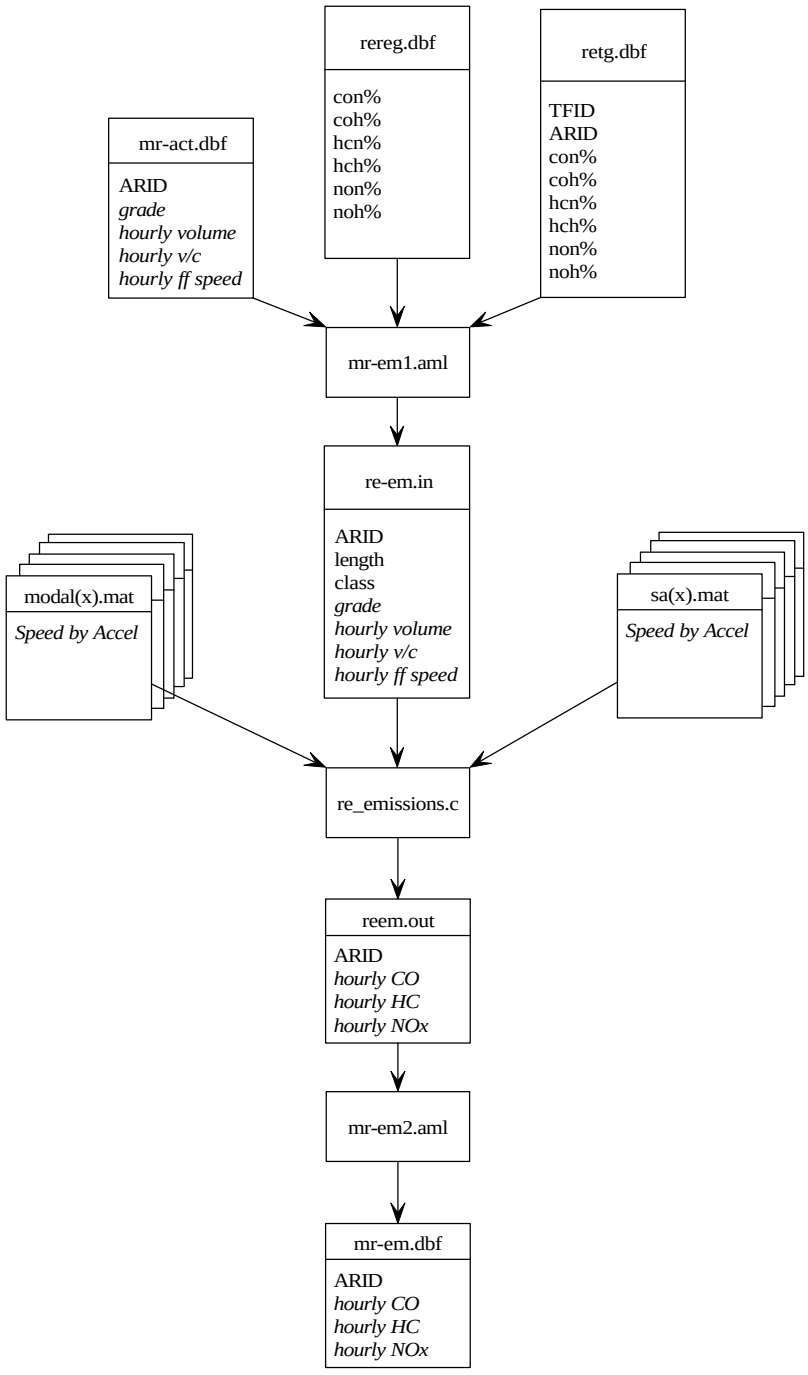

Figure 4.24 - Major Road Emissions Entities

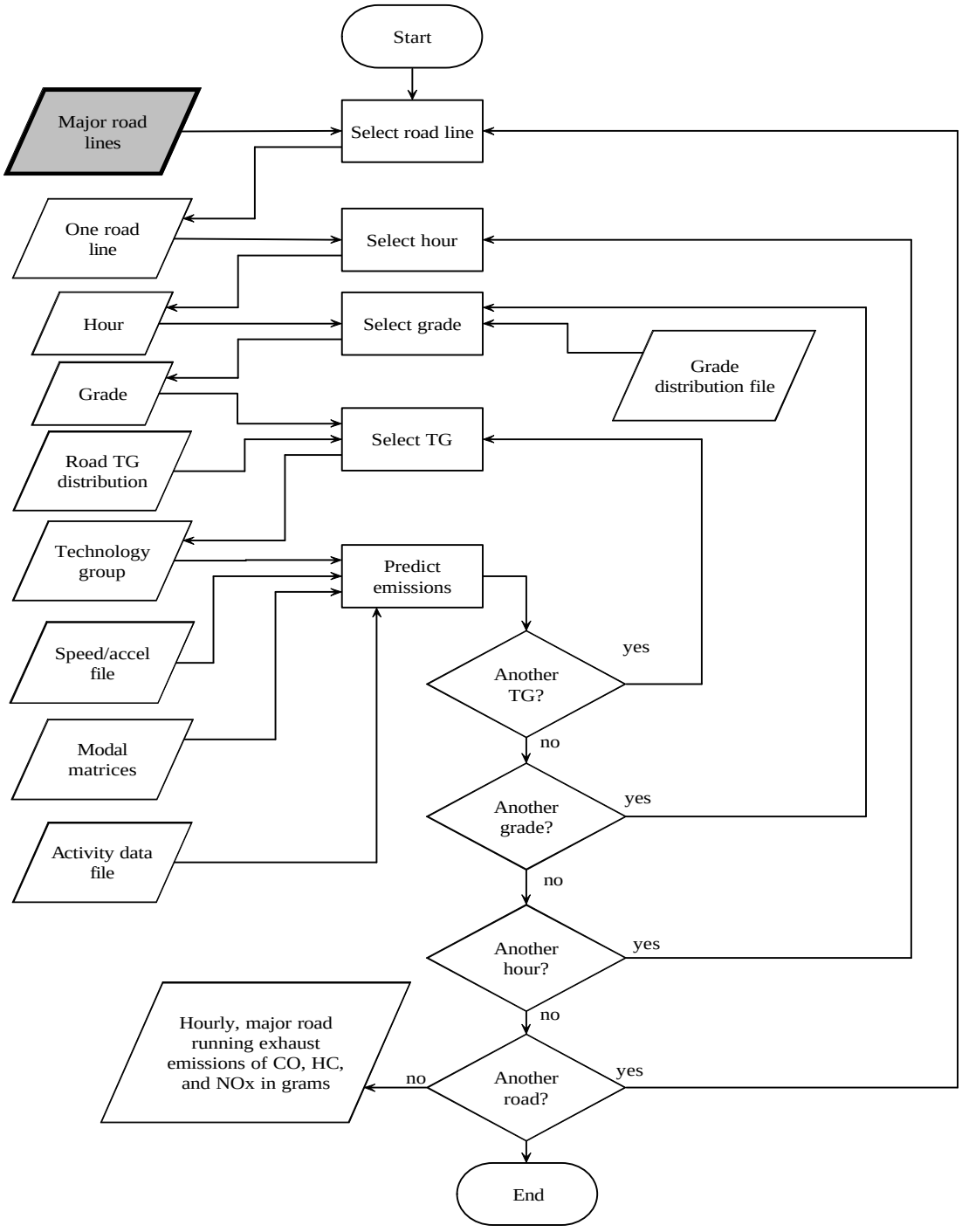

Figure 4.25 - Re_emissions.c Flow Chart

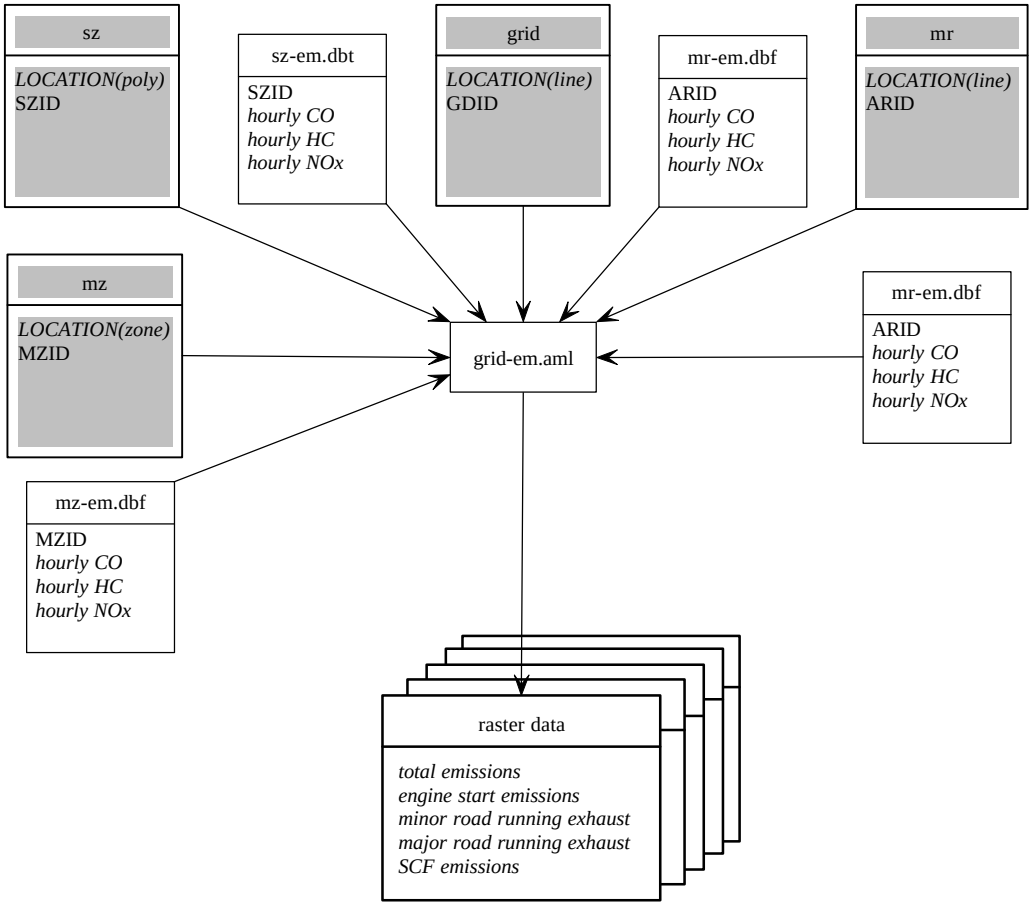

Figure 4.26 - Gridded Emissions Entities

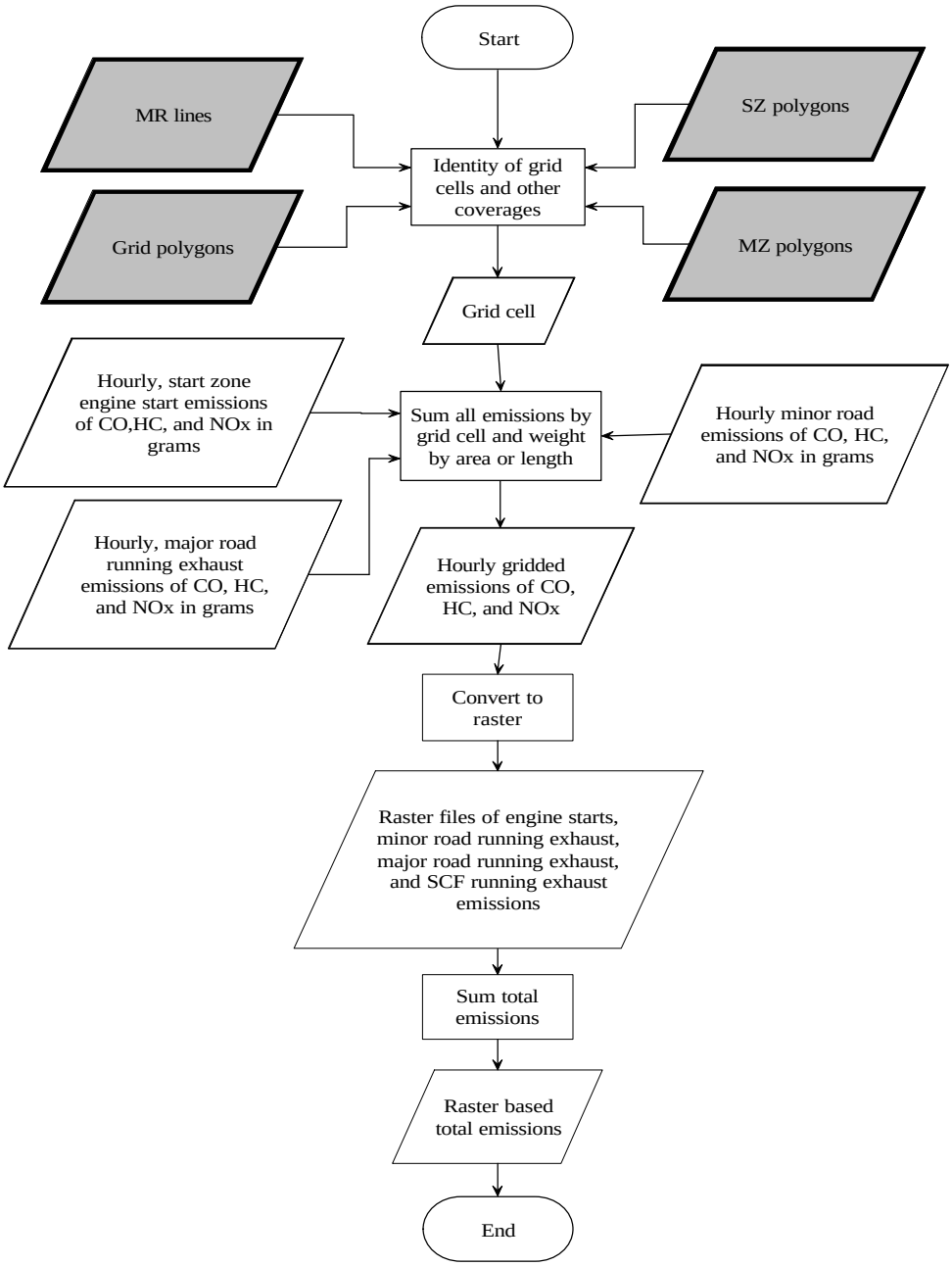

Figure 4.27 - *Grid-em.aml* Flow Chart